HANGMAN GAME

Xingrong Zong

Email: xz222bb@student.lnu.se

AAA

02.04.2020

Contents

1. Revision History

Date	Version	Description	Author
02.04.2020	1	Project plan: Provided a plan for the Hangman project in as much details as the developer and game designer can think right now. The project will be progressed based on the ideas of this project plan and giving further changes if necessary, such as customer requirements changing, adding or removing some functionalities due to time-consuming or staff ability.	Xingrong Zong
02.05.2020	2	Modelling + Software Design: Used diagrams tool to produce a view of project's structure and functions. Planned use cases for the project to give a	Xingrong Zong
13.07.2020	3	Test Plan: Plan the testing process and what parts of the project will be tested. And results of the testing on this project.	Xingrong Zong
11.08.2020	4	Final Project + Documentation: Reviewed all iterations and combined them to make a final report of this project	Xingrong Zong

2. General Information

Project Summary	
Project Name	Project ID
Hangman Game	20200202
Project Manager	Main Client
Xingrong Zong	Player
Key Stakeholders	
Project Manager Programmer / Game Designer Tester	Player
Executive Summary	
The project Hangman Game is to create a program selects a word and the player is going to guess the word by guessing letter after letter. For every wrong guess, the game will build a part of a man getting hanged. The number of wrongs that the player can have is ten. The player loses if there are no guesses left. The game also ends if the player identifies all the alphabets within ten tries, and the player will be considered a winner and can choose if he/she wants one more game round. The purpose of this Hangman project is to create an easy to build and friendly to play text-based game.	

3. Vision

The aim of the vision is to state the expected achievement of the project. It outlines the key goals to be accomplished.

The project Hangman Game is to create a word-guessing game. The game is written in Java.

For this text-based version, the player should be greeted with a menu to select either start or quit the game. When the game starts, a word from a predefined list of words will randomly be picked and the number of letters displayed with equally many underscore signs.

The system builds an “image” of the hanging man using the available characters, mostly are lines “|” or “-” on the keyboard. For every wrong guess, the game builds a part of a man getting hanged. The number of wrongs that the player can have is ten. If there are no guesses left, the player fails the task and can choose to start another round or quit the game. If the player manages to guess the word correctly within ten tries, the player will be considered a winner and will get the restart choice as well.

Reflection:

To create this vision document, the programmer and the game designer must understand the concept and rules of the hangman game. The project planning includes some additional functions for the player, such as to quit the game or restart the game.

4. Project Plan

4.1 Introduction

The project Hangman Game is to create a program in java code. The program is a text-based game starts with a predefined but not displayed word and the player is going to guess the word by guessing letter after letter. For every wrong guess, the game is building a part of a man getting hanged. The number of wrongs that the player can have is ten. The player loses if there are no guesses left. The game also ends if the player identifies all the alphabets within ten tries, and the player will be considered a winner with a score shows based on player's correct guesses.

4.2 Justification

The application is assigned to students who are participating in 1DV600 course as a project that aims to provide practical work with formal software project process.

4.3 Stakeholders

Project Manager: Carry out project planning and scheduling to verify whether the work meets the requirements. Keep track on the process and accomplishment stage of the project.

Programmer / Game Designer: Implement the code and designs functions to meet the functionalities requirements and project's requirements. Reflect the requirements that can not be accomplished due to skill, time or other factors.

Tester: Test the product's functions. Check if they meet the requirements or project planning and if there are any defects to develop.
Project Customer / End User: Player, people who will use the final product.

4.4 Resources

1DV600 Lectures (Videos)
Software Engineering by Ian Sommerville (Textbook)
Internet access

4.5 Hard- and Software Requirements

Specify what is used to develop and later run the software developed.
Game runner: Windows 10,
Java Version 13
IDE: IntelliJ 2020
JDK
JRE

4.6 Overall Project Schedule

First / Second/ Third / Forth iteration final deadline: 21/08/2020

4.7 Scope, Constraints and Assumptions

Scope: The game will be running as a java class in IDE IntelliJ 2020. The project is also text-based. The game can only be played by one player in one environment at a time. Player can choose to continue to play the game as many rounds as they want. Word is randomly selected from a predefined

vocabulary library in the same located package. The player has ten chances to guess the word before gets marked as failed.

Constraints: Due to time-consuming, developer's lack of practical skill and limited knowledge about developing and database, some preplanned functions cannot be developed for the project, such as user registration, multiplayer, time limit and the ability to delete letter.

Assumptions: The player has some degree of English vocabularies knowledge and understands the rules of the hangman game. The player can have a delightful experience.

5. Iterations

5.1 Iteration 1

Implement idea and some skeleton code for the project. Generate words for game

Estimated time: 4 h

Actual time: 7 h

5.2 Iteration 2

Add some features to the game. Using UML.

Estimated time: 5 h

Actual time: 15h

5.3 Iteration 3

Testing. Add additional features to the game in this iteration if needed.

Estimated time: 2 h

Actual time:10h

5.4 Iteration 4

Reiterate the steps in iteration 1 – 3 for a set of new features to make them as a whole project.

Estimated time: 1 h

Actual time: 1h

6. Risk Analysis

6.1 List of risks

Risk	Probability	Impact	Strategy
Underestimated development time	High	Serious	Wisely use time log to estimate developer's skills and time use. Plan the project more correctly and update the former if necessary.
Lack of knowledge	High	Tolerable	Gathering knowledge for the project from books and internet.
Requirements change	Moderate	Serious	Update the project plan as soon as possible, make sure it get solved before next change
Staff Emergency / Illness	Low	Serious	Try to do the work to save extra time for emergency. Almost no avoidance or minimization.
Hardware unavailability	Low	Serious	Save the code and plan work every time, and backup them on several

			platforms such as google drive and gitlab..

Reflection: I realize it is very important to make a preplanned project plan is the first step to start a project. It will save a lot of time and the probability that the risk will arise is reduced which increase productivity and efficiency. Preparation of minimizing the impact of them if they do occur and preparing for the worst and have a strategy in place to deal with it

7. Use Cases

Use Case 1: Start Game

Precondition: None

Postcondition: The game menu is shown

Main scenario:

1. Starts when the user wants to begin a session of the hangman game.
2. The system presents the main menu with a title, the option to play and quit the game.
3. The Gamer makes the choice to start the game.
4. The system starts the game (see Use Case 2)

Repeat from step 2

Alternative scenarios:

3.1 The Gamer makes the choice to quit the game.

1. The system quits the game (see Use Case 3)

4.1 Invalid menu choice

1. The system presents an error message

2. Go to step 2

Use Case 2: Play Game

Precondition: The game menu is shown

Postcondition: The game is running

Main scenario:

1. Starts when the user chooses to start the hangman game.
2. The system picks a random word from the predefined vocabulary library and shows the number of letters with underlines.
3. The Gamer guesses a correct letter.
4. The system adds the correct letter to the right position.
5. Repeat from step 3 until the word is guessed or there are no more attempts.
6. The system shows the message to tell whether the Gamer won or lost the game.
7. The Gamer choose to start a new game or quit the game.

Alternative scenarios:

3.1 Gamer's guess is not a letter

1. The system tells Gamer to guess a letter.
2. Go to step 3

7.1 The system shows menu

Use Case 3: Quit Game

Precondition: The game menu is shown

Postcondition: The game is terminated

Main scenario:

1. Starts when the user wants to quit the game.
2. The system prompts for confirmation.
3. The user confirms.
4. The system terminates.

Alternative scenarios:

3.1. The user does not confirm

1. The system returns to its precious state

8. Manual Test Cases

TC1.1: Start Game Successfully

Use case: UC 1 Start Game

Scenario: Start the game successfully

Description: The main scenario of UC1 is tested where a user starts game successfully.

Precondition: The only word option from Vocabulary library is “acquiesce”

Test Steps:

1. Run the project
2. System shows

“ Welcome to Hangman! Let's start!

1) Play Game

2) Quit Game”

3. Type 1 and press enter

Expected Results:

_ _ _ _ _ _ _ _ _

Wrong guesses:

Enter a letter (to quit type 2):

Actual Results: **Pass**

TC1.2: Quit Game Successfully

Use case: UC 1 Start Game

Scenario: Quit the game successfully

Description: The main scenario of UC1 is tested where a user quits game successfully.

Precondition: The only word option from Vocabulary library is “acquiesce”

Test Steps:

1. Run the project
2. System shows

“ Welcome to Hangman! Let's start!

1) Play Game

2) Quit Game”

3. Type 2 and press enter

Expected Results:

Process finished with exit code 1

Actual Results: **Pass**

TC2.1: Invalid input and promote input again

Use case: UC 2 Play Game

Scenario: The player types in invalid things

Description: The main scenario of UC2 is tested where a user types in invalid answers.

Precondition:

The game is running.

The only word option from Vocabulary library is “acquiesce”

Test Steps with Expected Results:

1. System shows the underlines as the number of letters of word:

_ _ _ _ _ _ _ _ _

Wrong guesses:

Enter a letter (to quit type 2):

2. Type “5” and press enter.
3. System checks, the input is not a letter and shows:

   Please enter a letter.

4. Repeat from step 2 until the input is letter:

The word was acquiesce.

Actual Results: **Pass**

TC2.2: Player wins the game without any wrong guess

Use case: UC 2 Play Game

Scenario: The player wins the game without any wrong guess

Description: The main scenario of UC2 is tested where a user types in invalid answers.

Precondition:

The game is running.

The only word option from Vocabulary library is “acquiesce”

Test Steps with Expected Results:

1. System shows the underlines as the number of letters of word:

_ _ _ _ _ _ _ _ _

Wrong guesses:

Enter a letter (to quit type 2):

2. Type “a” and press enter.
3. System checks, the letter is correct and shows on the corresponding position.

a _ _ _ _ _ _ _ _

Wrong guesses:

Enter a letter (to quit type 2):

4. Repeat from step 2 (c, q, u, i, e, s):

a c q u i e s c e

YOU SUCCEED

The word was acquiesce.

Actual Results: **Pass**

TC2.3: Player loses the game without any correct guess

Use case: UC 2 Play Game

Scenario: The player loses the game without any correct guess

Description: The main scenario of UC2 is tested where a user types in invalid answers.

Precondition: The only word option from Vocabulary library is “acquiesce”

Test Steps with Expected Results:

1. System shows the underlines as the number of letters of word:

_ _ _ _ _ _ _ _ _

Wrong guesses:

Enter a letter (to quit type 2):

2. Type “b” and press enter.
3. System checks, the letter is correct and shows on the corresponding position.

_ _ _ _ _ _ _ _ _

Wrong guesses: b

Enter a letter (to quit type 2):

4. Repeat from step 2 nine more times with wrong guess:

_ _ _ _ _ _ _ _ _

|----------

| |

| 0

| /|\

| / \

|

|

____|____

YOU DIED

The word was acquiesce.

Actual Results: **Pass**

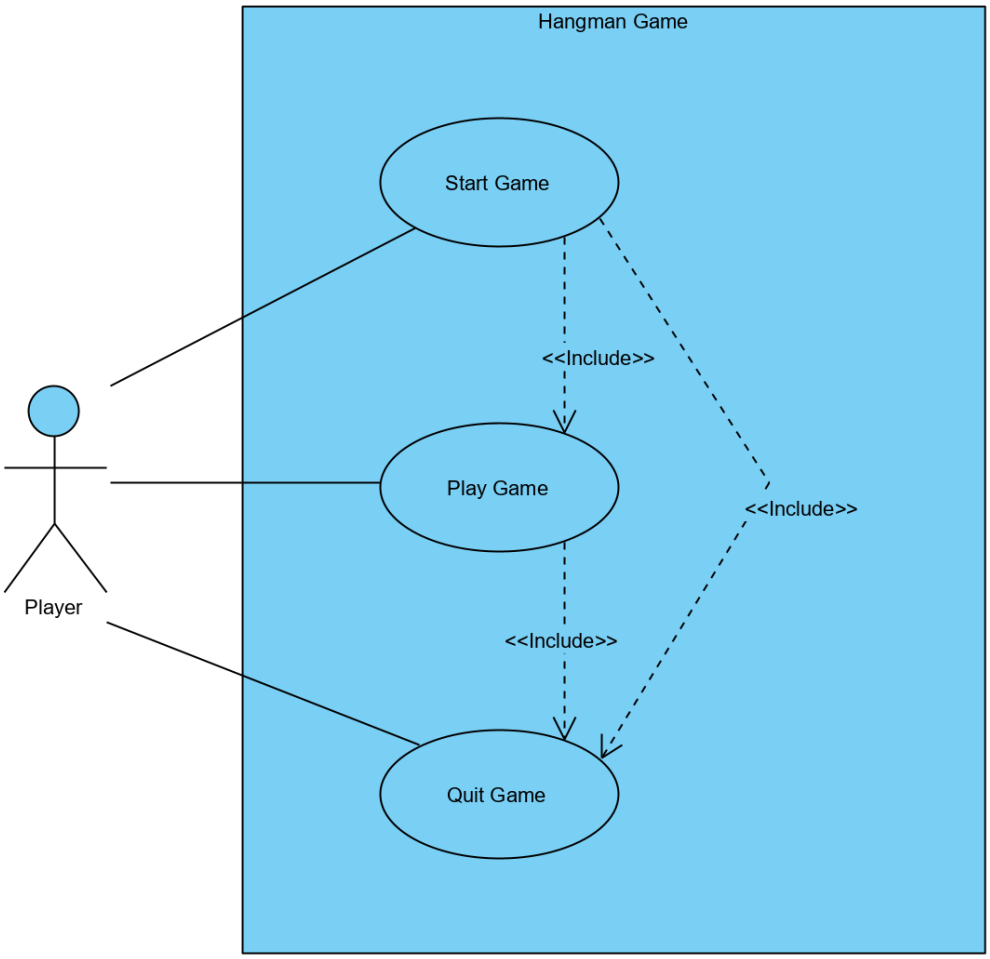
Test Report:

Test	UC1	UC2
TC1.1	1/OK	0
TC1.2	1/OK	0
TC2.1	0	1/OK
TC2.2	0	1/OK
TC2.3	0	1/OK

9. UML Diagrams

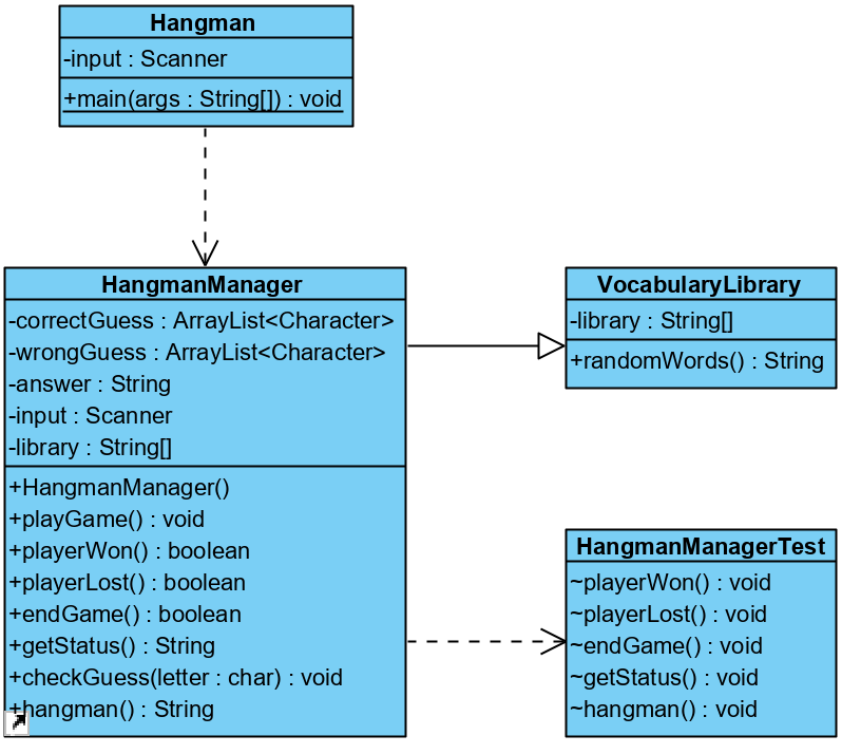
9.1 Use Case Diagram

The use case diagram is to capture the core functionalities of the project system that will turn into design and development decisions.



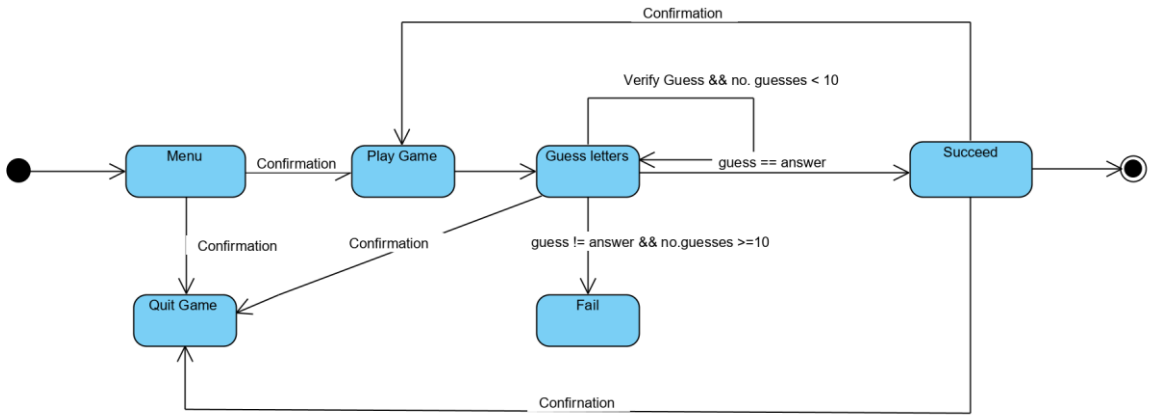
9.2 Class Diagram

The class diagram describes the structure of the project system by showing the classes, methods and relations.



9.3 State Machine Diagram

The state machine diagram captures the dynamic behaviour of the project system and describes the different states of the project system and its entities.



10. Time log

Task	Estimated Time	Actual Time	Time Difference
Create a Project Plan	2.5h	3.5h	1h
Update the Project Plan	30min	30min	0
Create a Time Log Table	10min	25min	15min
Update the Time Log table	5min	5min	0
Create a Use Cases Diagram	25min	35min	10min
Update the Use Cases Diagram			
Create the fully dressed Use Cases	1.5h	2.5h	1h
Update the fully dressed Use Cases			
Create a State Machine Diagram	40min	100h	1h
Update the State Machine Diagram			
Create a Class Diagram	25min	55min	30min

Update a Class Diagram			
Read materials	6h	8h	2h
Update materials			
Implement the code for the Hangman Game	6h	11h	5h
Update the code	1h	2h	1h
Create Test Plan	1h	1h	0
Create Manual Test	2h	2.5h	0.5h
Create Unit Test	2h	2h	0
Reflection	20min	20min	0